8-2003

A Study on the Prediction of Viscosity with the use of Numerical Simulation Software

Brian Robinson
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Brian Robinson entitled "A Study on the Prediction of Viscosity with the use of Numerical Simulation Software." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Engineering Science.

Dr. Basil Antar, Major Professor

We have read this thesis and recommend its acceptance:

Dr. Frank Collins, Dr. Gary Flandro

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Brian Robinson entitled "A Study on the Prediction of Viscosity with the use of Numerical Simulation Software." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Engineering Sciences.

Dr. Basil Antar
Dr. Basil Antar, Major Professor

We have read this thesis
and recommend its acceptance:

Dr. Frank Collins
Dr. Frank Collins

Dr. Gary Flandro
Dr. Gary Flandro

Accepted for the Council:

Dr. Anne Mayhew
Vice Provost and Dean of Graduate Studies

(Original signatures are on file with official student records)

A Study on the Prediction of Viscosity with the use of Numerical Simulation Software

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Brian Christopher Robinson
August 2003

Abstract

A study was undertaken to numerically simulate the flow of highly viscous fluid. Simulating a highly viscous flow within a parallel plate plastometer will hopefully begin to bridge the gaps in the current knowledge of these viscous models. The modeling of this flow was performed using the canned code FiDAP from Fluent Inc. The results are compared with the analytical results calculated by NASA Marshall researchers.

Table of Contents

List of Figures

Nomenclature

Symbol	Meaning
ρ	density
η, μ	viscosity
F	Force
f	arbitrary function
h_o	initial height of specimen
P	Pressure
P_a	atmospheric pressure
R	radius
r	radial position
t	time
V	volume
v	velocity of the plate
v_z	axial velocity
v_r	radial velocity
z	axial position within the specimen

Chapter 1 - Introduction

1.1 - Background

For many years there have been efforts all over the world to find better ways of calculating the viscosity of fluids. As a key component to the momentum equation it is necessary to have an accurate model of the viscosity over a range of temperatures. The problem is that there isn't a method that will encompass all temperatures. Some methods work well for high temperatures, some for low temperatures, and others in the mid range. The parallel plate plastometer and the method used to find the viscosity indirectly from it are supposed to help bridge the gap between the mid and high range temperature fluids. The purpose of the research discussed in this paper was to use data generated by a parallel plate plastometer and create a numerical simulation that can aid in the accurate prediction of the viscosity of a particular fluid. Currently the prediction of viscosity is based on an analytical solution. In the numerical case a numerical software program called FiDAP will be used to simulate the flow between two parallel plates aligned horizontally in which the upper plate is subjected to a prescribed force while the lower plate is held stationary. This type of configuration is what is known as the parallel plate plastometer. The primary interest in simulating the plastometer is that it may be possible to generate better estimates of the viscosity of fluids using numerical methods rather than analytical methods.

In order to perform this study several important steps had to be followed. The first step was to determine how to use FiDAP in order to accurately simulate the

compression of a fluid bridge between two parallel plates. The next step was to compare the simulation with an analytical model to determine if the simulation was accurate. The final step in the process was to use data collected from the plastometer and create a simulation. From the simulation we would collect pressure data which, through some data reduction, would predict the viscosity of the fluid.

In this study the simulation represents a parallel plate plastometer which compresses a fluid bridge between two parallel plates. As the plates are compressed the fluid moves both axially and radially, but it does not move with any rotation about the axis. This allows the axi-symmetric condition to be applied. Some other assumptions and conditions that will be assumed are that the fluid is Newtonian as well as that the no-slip condition holds true.

The simulation detailed in this study consists of a highly viscous liquid bridge between two parallel plates which are being compressed. The simulation will be run using several different values of viscosity. Pressure values will be collected throughout the simulation and then integrated over the surface area, giving a value for the force applied to the plate, thereby creating a force vs viscosity graph. Using the information about the force applied in the actual experiment on the plastometer it is possible plot an intersecting line that will predict the viscosity of the fluid.

1.2 - Fill Method Background

Some research has already been performed in this area using the numerical code FiDAP. This work, completed by Hlady[2], employed a slightly different method than

will be described in this paper. This method is called the ‘fill’ method. The fill method is a volume tracking technique in which each element of the mesh is expressed as a fraction denoting the concentration of fluid within the element. An element with a fractional fill equal to zero is empty while an element with a fractional fill equal to one is full. Any partially full element will have a fraction between zero and one.[2] This method requires that the free surface of the fluid be within the mesh. It cannot be located at the edge of the mesh in which it is to expand. As described by Hlady[2] this method gives an inexact representation of the surface, so it has been decided to proceed using a ‘free surface’ method which will be described later.

Chapter 2 - Parallel Plate Plastometer

The parallel plate plastometer is a large, but very simple machine. It's only function is to find the deformation rate of a fluid at or above the ambient temperature. Here the parallel plate plastometer is used to indirectly measure the viscosity of fluids by applying a known pressure to the top plate and measuring the change in separation, h, of the two plates with respect to time. As shown in Figure 2.1 there are two plates, top(1) and bottom(2), a dial(3) for measuring the separation of the two plates, two movable weights(4,7), a lever arm(5), a window and hatch(6) for viewing and adjusting the specimen, the shaft(8) and the connection to the top plate(9), a turntable(10), and the stand itself(11). Parts 1, 2, 9, 10, and 11 are all enclosed within the oven. With the following information [3]

$$\text{Area} = p \cdot D^2 / 4 \tag{2.1}$$

$$h = (h_o + h_f) / 2 \tag{2.2}$$

$$s = 50000\,\text{Pa} \tag{2.3}$$

$$F = s \cdot \text{Area} \tag{2.4}$$

$$V = h_{o_i} \cdot \text{Area} \tag{2.5}$$

and the Fontana equation[6], to be derived later,

$$h = \frac{2 \cdot p \cdot F \cdot (h)^5}{3 \cdot V \cdot \text{rate} \cdot [2p \cdot (h)^3 + V]} \tag{2.6}$$

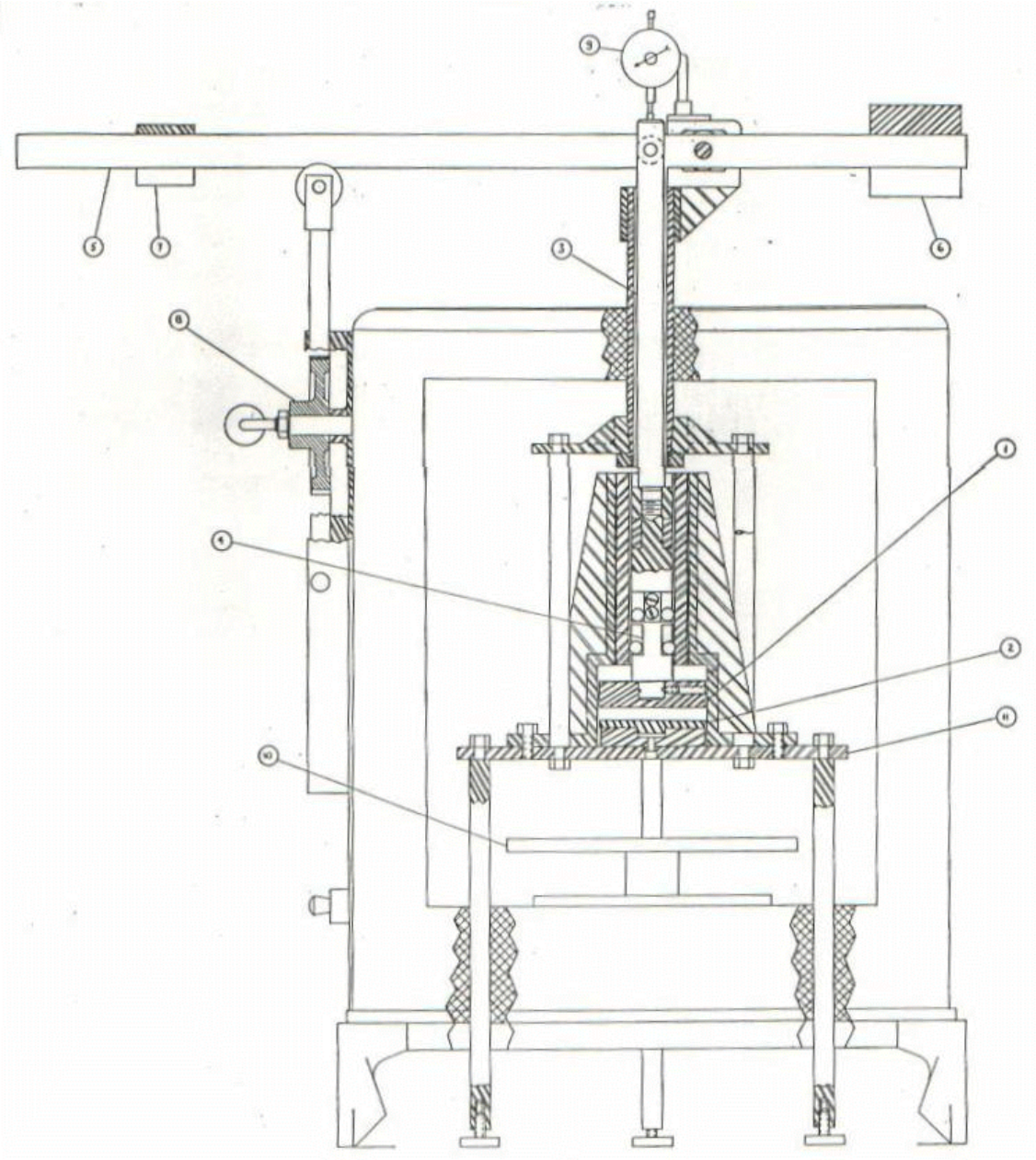

Figure 2.1 - Parallel plate plastometer

Area - Area of cylindrical sample fluid
D - Diameter of sample
h_o - initial height of the sample
h_f - final height of the sample
h - average height of sample
σ - Pressure applied to top plate
F - Force applied to top plate
V - Volume of sample
η - Viscosity of sample
rate - rate of change in separation of the plates

the rate and therefore the viscosity of the fluid can be generated experimentally using the parallel plate plastometer. The following, as described by Dienes and Klemm[3], is the procedure for using a parallel plate plastometer.

1. The oven will be heated to the desired temperature for approximately one hour to ensure equilibrium within the apparatus.
2. Weights are positioned on the lever arms applying the desired load.
3. The dial gauge is set to minus two divisions (one division = .001 inch) when the plates are together under test load (no specimen). This corrects for the thickness of the two one-mil sheets of cellophane which are place on the two ends of the specimen, and fixes the zero point for the specimen height readings
4. Center the specimen (weighed to the nearest milligram) between two 4x4x0.001 in. pieces of plain cellophane, and insert this assembly centrally between the parallel plates. If preheating is necessary, this may be done in the plastometer or in the special clamps provided.
5. Apply the test load and read the separation, h, of the two parallel plates on

the dial gauge to an accuracy of +/- 0.0002 (0.2 divisions) as a function of time.

6. Plot on linear coordinate paper, as the test proceeds, the reciprocal of the fourth power of the plate separation, as ordinate against the time in seconds.

7. Continue the test until a straight line of slope is defined in the range of h corresponding to R: h=10 (R = radius of the specimen) and h = 10 divisions. Loads and temperatures must be adjusted to satisfy the requirements.

8. Calculate the viscosity from the following equation:

$$h = 8.21 \times 10^{6} (W / mV^{2}) \qquad (2.7)$$

where h= viscosity, poises, W = applied load, kilograms, m = slope of plot of $1/h^4$ vs time, $cm^{-4}sec^{-1}$, and V = volume, obtained from specimen mass and density, cm^3.

Equation 2.7 is of an older and less accurate analytical model dated to 1947 or before and therefore is slightly different from the currently used Fontana equation[6]. The computer simulation will hopefully predict an even more accurate viscosity of these fluids than the analytical solution currently used.

Chapter 3 - Analytical Solution

The best method for initially testing our findings from the simulation would be to compare the numerical results to an analytical model. In order to find a generalized solution to the problem several equations were derived using an example problem set up similar to the plastometer described in Bird, Armstrong. and Hassager[1].

Starting from the continuity equation and the momentum equations, assuming there is no flow in the theta direction, in cylindrical form they become:

$$\frac{\partial r}{\partial t} + \nabla \cdot \left(r \bar{v} \right) = 0, \tag{3.1}$$

$$r\left(\frac{\partial v_r}{\partial t} + v_r \frac{\partial v_r}{\partial r} + \frac{v_q}{r}\frac{\partial v_r}{\partial q} - \frac{v_q^2}{r} + v_z \frac{\partial v_r}{\partial z} \right) =$$

$$m\left[\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r v_r \right) \right) + \frac{1}{r^2}\frac{\partial^2 v_r}{\partial q^2} + \frac{\partial^2 v_r}{\partial z^2} - \frac{2}{r}\frac{\partial v_q}{\partial q} \right] - \frac{\partial p}{\partial r} + r g_r, \text{ and} \tag{3.2}$$

$$r\left(\frac{\partial v_z}{\partial t} + v_r \frac{\partial v_z}{\partial r} + \frac{v_q}{r}\frac{\partial v_z}{\partial q} + v_z \frac{\partial v_z}{\partial z} \right) =$$

$$m\left[\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r v_z \right) \right) + \frac{1}{r^2}\frac{\partial^2 v_z}{\partial q^2} + \frac{\partial^2 v_z}{\partial z^2} \right] - \frac{\partial p}{\partial z} + r g_z. \tag{3.3}$$

Making several assumptions these equations can be reduced significantly. These assumptions include: constant density, quasi-steady state, no gravitational forces, and that flow will be mainly in the r-direction. The following terms can be dropped as a result

$$\frac{\partial r}{\partial t} \to 0 \qquad \frac{\partial v_r}{\partial t} \to 0$$

$$\frac{\partial v_q}{\partial q}, \frac{\partial^2 v_r}{\partial q^2}, \frac{\partial v_r}{\partial q}, \frac{v_q^2}{r} \to 0$$

$$r g_r, r g_z \to 0$$

$$v_z << v_r \quad \& \quad \frac{\partial v_r}{\partial r} << \frac{\partial v_r}{\partial z}$$

$$\therefore \quad v_r \frac{\partial v_r}{\partial r}, v_z \frac{\partial v_r}{\partial z}, \frac{\partial^2 v_r}{\partial r^2} \to 0.$$

By dropping the above terms, the continuity and momentum equations become:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r v_r\right) + \frac{\partial v_z}{\partial z} = 0, \tag{3.4}$$

$$0 = -\frac{\partial p}{\partial r} + \mu \frac{\partial^2 v_r}{\partial z^2}, \text{ and} \tag{3.5}$$

$$0 = -\frac{\partial p}{\partial z}. \tag{3.6}$$

The continuity equation then demands that

$$v_r = rf(z,t). \tag{3.7}$$

Taking the partial derivative of the *r*-momentum equation with respect to *z* and the partial derivative of the *z*-momentum equation with respect to *r* as shown Eq. 3.8 and 3.9 result.

$$0 = -\frac{\partial^2 p}{\partial r \partial z} + \mu \frac{\partial^3 v_r}{\partial z^3} \tag{3.8}$$

$$0 = -\frac{\partial^2 p}{\partial r \partial z} \tag{3.9}$$

So then

$$\frac{\partial^3 v_r}{\partial z^3} = 0 \tag{3.10}$$

and the integral with respect to z is a constant. Therefore

$$\frac{\partial^2 v_r}{\partial z^2} = const = \rho \frac{\partial^2 f}{\partial z^2} \tag{3.11}$$

and

$$\frac{\partial p}{\partial r} = \mu \rho c_2 . \tag{3.12}$$

Integrating with respect to *r*, the following equation emerges

$$p = \frac{1}{2} \mu \rho c_2 r^2 + c_o \rightarrow p = p_2 r^2 + p_o . \tag{3.13}$$

By substituting $v_r = rf(z,t)$ and Equation 3.13 into the reduced form of the continuity and momentum equations the following two equations result

$$2f + \frac{\partial v_z}{\partial z} = 0 \tag{3.14}$$

$$-2P_2 + \mu \frac{\partial^2 f}{\partial z^2} = 0 \tag{3.15}$$

as well as the following boundary conditions:

$$\frac{\partial f}{\partial z} = 0 \text{ at } z = \frac{h}{2};\ f = 0 \text{ at } z = h$$

$$v_z = 0 \quad \text{at} \quad z = 0; \quad v_z = dh/dt \quad \text{at} \quad z = h$$

$$P = P_a \text{ at } r = R^2.$$

Following some algebra Eq. 3.15 becomes

$$\frac{2P_2}{\mu} = \frac{\partial^2 f}{\partial z^2}. \tag{3.16}$$

Integrating both sides with respect to *z*, results in

$$\frac{\partial f}{\partial z} = \frac{2P_2}{\mu} z + C_1. \tag{3.17}$$

The boundary conditions can then be applied to determine the constant of integration, C_1,

$$C_1 = -\frac{P_2 h}{\mu} \tag{3.18}$$

Integrating a second time with respect to *z*, Eq 3.19 is obtained

$$f = \frac{P_2 z^2}{\mu} + C_1 z + C_2 \tag{3.19}$$

After applying the second boundary condition, $f = 0$ at $z = h$, it can be found that

$$0 = \frac{P_2 h^2}{\mu} - \frac{P_2 h^2}{\mu} + C_2 \tag{3.20}$$

where the constant of integration, C_2, is

$$C_2 = 0.$$

This results in the following equation:

$$f = \frac{P_2}{\mu} z^2 - \frac{P_2 h}{\mu} z \tag{3.21}$$

Substituting f into Eq. 3.14

$$2\left[\frac{P_2}{\mu} z^2 - \frac{P_2 h}{\mu} z\right] = -\frac{\partial v_z}{\partial z} \tag{3.22}$$

and then integrating both sides with respect to *z*,

$$-\frac{2P_2}{3\mu} z^3 + \frac{P_2 h}{\mu} z^2 + C_3 = v_z \tag{3.23}$$

After applying the third boundary condition, $v_z = 0$ at $z = 0$, it can be found that

$$C_3 = 0.$$

So then the following equation exists for v_z

$$v_z = -\frac{2P_2}{3\mu} z^3 + \frac{P_2 h}{\mu} z^2 \tag{3.24}$$

Using the boundary condition $v_z = dh/dt$ at $z = h$, then

$$dh/dt = -\frac{P_2 h^3}{3\mu} \tag{3.25}$$

and

$$\frac{P_2}{\mu} = \frac{3\,dh/dt}{h^3}\ . \tag{3.26}$$

Substituting Eq. 3.26 into those for v_z and f, they can be simplified to the following two equations:

$$v_z = dh/dt\left[3\left(\frac{z}{h}\right)^2 - 2\left(\frac{z}{h}\right)^3\right] \tag{3.27}$$

$$f = \frac{3\,dh/dt}{h}\left[\frac{z^2}{h^2} - \frac{z}{h}\right] \tag{3.28}$$

Remembering that $v_r = rf(z,t)$, then

$$v_r = \frac{3r\,dh/dt}{h}\left[\left(\frac{z}{h}\right)^2 - \frac{z}{h}\right]. \tag{3.29}$$

After finding the equations for the axial velocities it becomes possible to determine the pressure. As stated previously

$$P = P_o + P_2 r^2\ . \tag{3.30}$$

It was also stated earlier in Eq. 3.25 that

$$\frac{P_2}{\mu} = \frac{3\,dh/dt}{h^3}\ .$$

Now substitute the above for P_2 in Eq. 3.30 results in

$$\frac{P - P_o}{r^2} = \frac{3\mu\,dh/dt}{h^3} \text{ or } P - P_o = \frac{3\mu\,dh/dt\,r^2}{h^3}\ . \tag{3.31}$$

After applying the condition $P = P_a$ at $r = R^2$, one can find that

$$P_o = P_a - \frac{3\mu \, dh/dt \, R^2}{h^3}. \tag{3.32}$$

Which then means that

$$P - P_a = \frac{3\mu(-dh/dt)R^2}{h^3}\left[1 - \frac{r^2}{R^2}\right]. \tag{3.33}$$

The force and consequently the viscosity can be obtained by integrating $P - P_a$ with respect to *r* and q from zero to *R*, where is *R* constant.

$$F = \frac{3p \, R^4 h\left(-dh/dt\right)}{8h^3} \tag{3.34}$$

Solving the above differential equation for *h(t)* leads to the equation for the viscosity as shown.

$$h = \frac{16F}{3p \, R^4}\left[\frac{t}{\frac{1}{h^2} - \frac{1}{h_o^2}}\right] \tag{3.35}$$

where h_o is the initial height and *h* is the final height at time *t*. In order to derive the Fontana Equation, a small deviation from the above derivation must be made first. If the integration performed between Eq. 3.33 and 3.34 is such that *R* is a function of time, the following force and viscosity equations result [3]

$$F = \frac{3h\,V^2\left(-dh/dt\right)}{2p\,h^5} \tag{3.36}$$

$$h = \frac{8p\,F}{3V^2}\left[\frac{t}{\frac{1}{h^4} - \frac{1}{h_o^4}}\right] \tag{3.37}$$

As described in Fontana[6] when a species is compressed axially two forces result. These are the vertical compressive force, F_v, and the horizontal force, F_H, which keeps the specimen in a cylindrical from during the flow. These forces are

$$F_{\mathrm{v}} = \frac{3hV\dfrac{dh}{dt}}{h^3} \tag{3.38}$$

$$F_H = \frac{3hV^2\dfrac{dh}{dt}}{2p\,h^5} \tag{3.39}$$

Adding these two equations together results in the Fontana equation which is currently being used by researchers at NASA Marshall in their experiments.

$$h = \frac{2\cdot p\cdot F\cdot (h)^5}{3\cdot V\cdot \dfrac{dh}{dt}\cdot\left[2p\cdot (h)^3 + V\right]} \tag{3.40}$$

These solutions can now be graphed along with the data collected from the simulation in order to check for correlations. The numerical solution and the analytic solution will not match exactly, but should be close, as shown in Figures 3.1 and 3.2. This is due to the fact that the analytical solution does not account for the bulge that forms when the liquid bridge is compressed. Figure 3.1 shows the radial velocity of the fluid at the exposed free surface for both the analytical and numerical solutions at the same time value. While Figure 3.2 shows the force in Newtons versus the radius for the analytical solution as well as for two nodal lines. These nodal lines are at the center of the liquid bridge as well as the second nodal line from the top plate. It can already be seen that the values at the centerline of nodes gives what appears to be more accurate results than the nodal line closest to the top plate. In these diagrams, since the analytical solution assumes there is always a cylindrical bridge, it does not allow for any deformation of the surface.

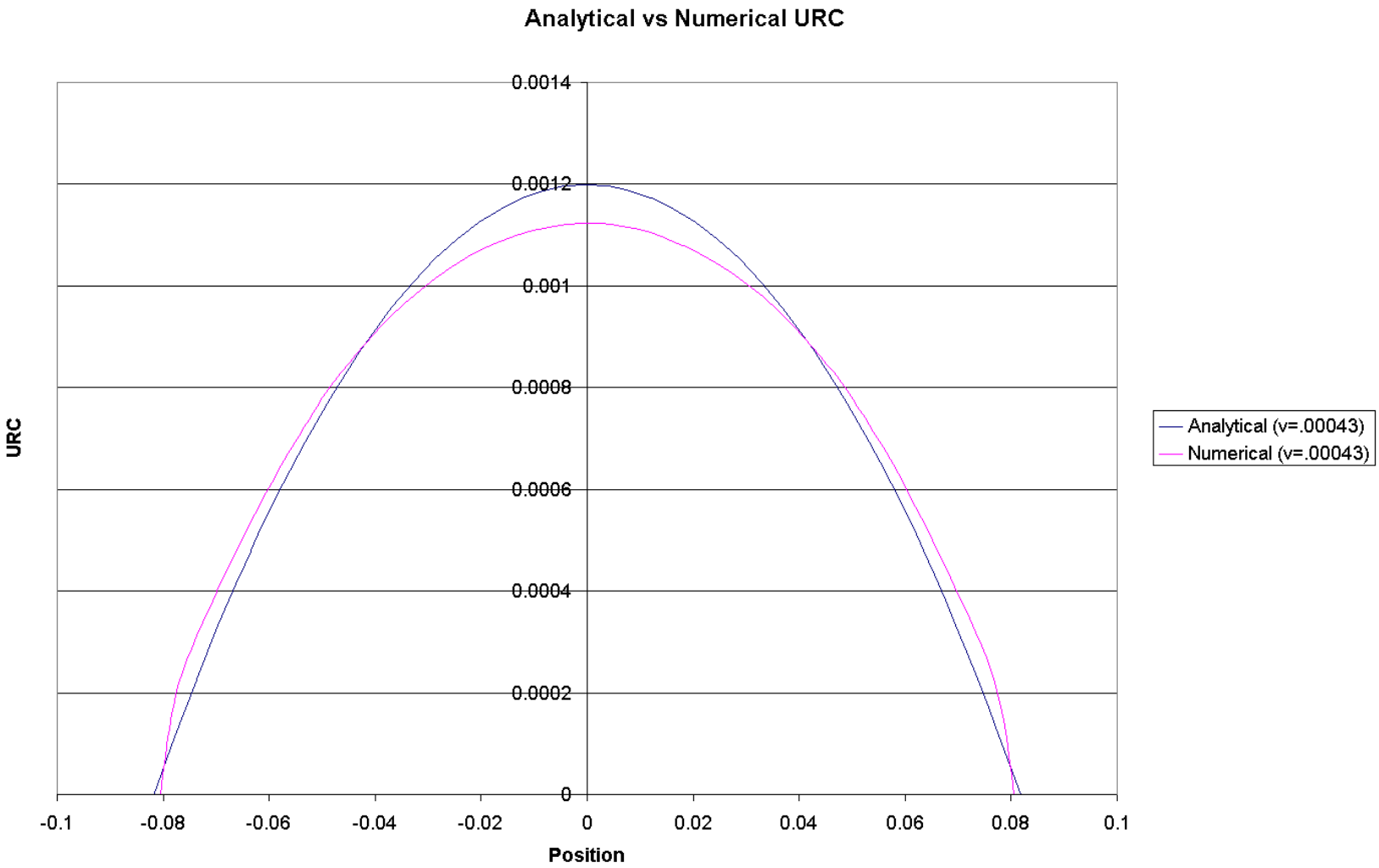

Figure 3.1 - Analytical vs numerical velocity profile at the free surface

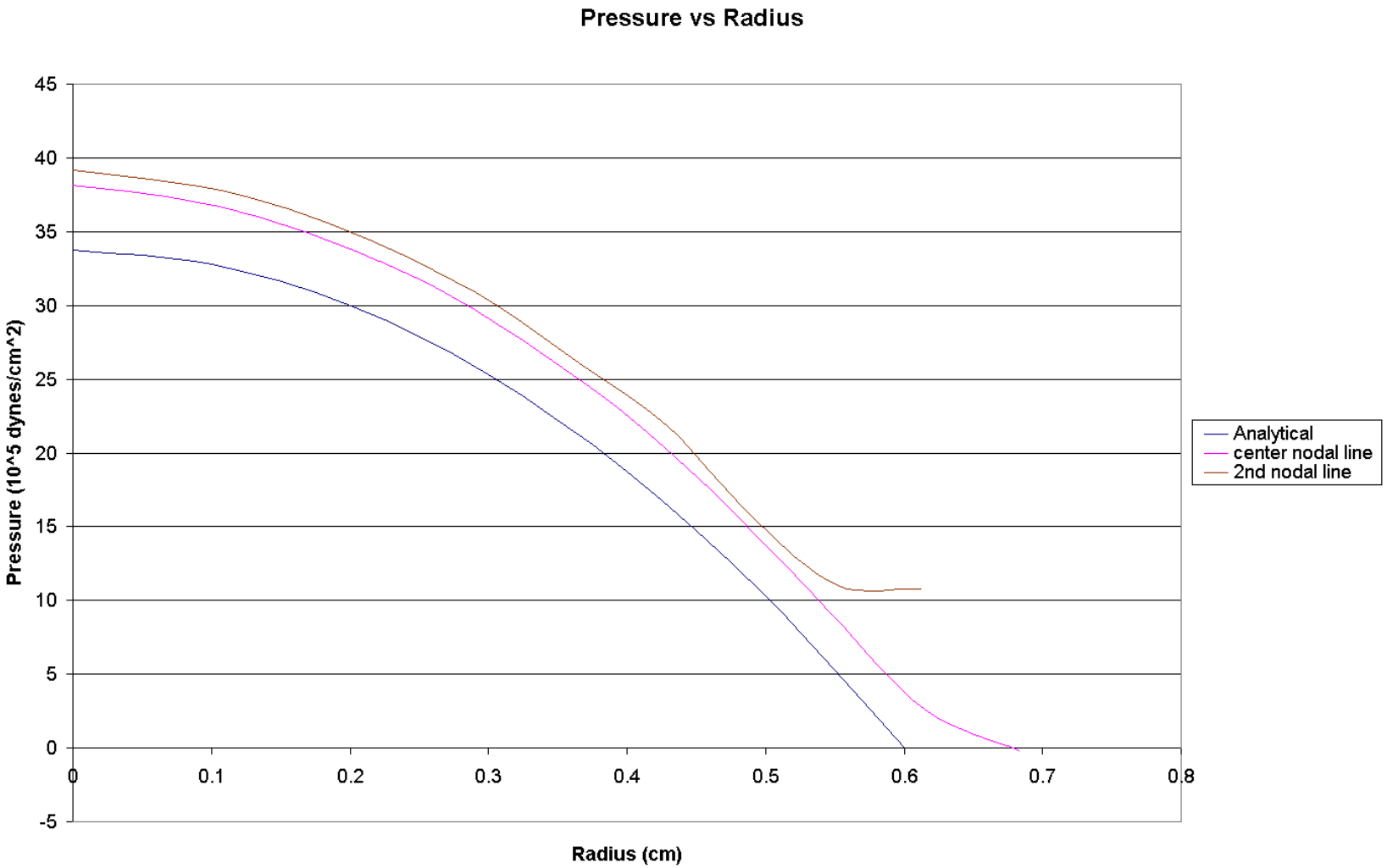

Figure 3.2 - Pressure vs. radius graph showing the analytical as well as the numerical solution for two nodal lines

Chapter 4 - Numerical Analysis

4.1 - Description of FiDAP

FiDAP is a very powerful and versatile computational fluid dynamics program that can be used to solve a wide range of problems. FiDAP uses the Finite Element Model (FEM) in order to simulate most flows. This gives it the ability to separate the fluid flow into small regions when solving the equations of motion. With the ability to automatically generate a finite element mesh given the constraints of the problem it became relatively easy to generate the conditions and geometry of this problem.

The program is broken down into eight different modules which carry out functions during the different parts of a simulation. The simulation focused on in this study requires the following modules: FiGEN, FiPREP, FiPOST, and FiSOLV. The first module, FiGEN, allows the generation of both simple and complex mesh designs for the simulation. FiPREP has the ability to perform several functions. FiDAP will use this module to specify the type of problem, the equations that are needed to be solved, the procedures for finding the solution, properties of the fluid and boundaries, as well as boundary conditions. The FiSOLV module runs the code, solving the conservation equations creating the simulation. This module transforms the governing partial differential equations in to algebraic equations and then finds a solution to those equations. The final module that will be employed is the FiPOST module. As with all of the modules, FiDAP's graphical user interface or command prompt can be utilized. The command prompt is the most time effective choice for data collection. This module via

command prompt allows pressure, velocity, position data to be collected quickly and easily.

4.2 - Simulation with FiDAP

The first step in solving this problem, with the help of FiDAP, is to use its mesh generation package FiGEN. With this package the size and shape of the object can be input as well as generating an appropriate mesh. The mesh will become more important later when the work begins on the final design to compare numerical, analytical, and experimental results. For now, a generic mesh will be used that is has nodal lines evenly distributed across the object. At this point in the process, the input file will resemble Figure 4.1. The first two lines open and set the defaults for the module. Lines three through six tell the module to create points at the specified coordinates. In the future these values will be defined as variables, *$x* and *$y*, as seen in the Appendix A-1 and A-2. Lines seven through twenty-two create the perimeter lines connecting each of the four corners. The mesh is created in lines twenty-three through twenty-six by first selecting each line segment creating a closed loop (it is important that it is a closed loop) and defining it with the “mface” command or the “mesh face” command. After the face is defined the nodal lines are created with the “medge” or “mesh edge” command. Within these commands it is necessary to define the number of nodal lines as well as their orientation, equally spaced or dense areas of nodal lines, such as near the wall. Figure 4.2 shows a graphical representation of the input file so far. FiDAP recognizes the vertical direction as the r-axis and the horizontal direction as the z-axis, so the simulation must be

```
FI-GEN ( ELEMENT=1,POINT=1,CURVE=1,SURFACE=1,NODE=0,MEDGE=1,MLOOP=1,MFACE=1,
BEDGE=1,SPAVE=1,MSHELL=1,MSOLID=1,COORDINATE=1 )
point(coordinates,x=0,y=0)
point(coordinates,x=.249,y=0)
point(coordinates,x=0,y=.51)
point(coordinates,x=.249,y=.51)
point(select,id,window=1)
1
2
curve(line,showlabel)
point(select,id,window=1)
3
4
curve(line,showlabel)
point(select,id,window=1)
1
3
curve(line,showlabel)
point(select,id,window=1)
2
4
curve(line,showlabel)
curve(select,id,window=1)
1
3
2
4
mface(wireframe,edg1cnt=1,edg2cnt=1,edg3cnt=1,edg4cnt=1)
curve(select,id,window=1)
2
1
medge(frstlast,intervals=20,ratio=0.000000,2ratio=0.000000,pcentr=0.000000)
curve(select,id,window=1)
3
4
medge(frstlast,intervals=10,ratio=0.000000,2ratio=0.000000,pcentr=0.000000)
mface(select,id,window=1)
1
mface(mesh,map,entity="fluid")
medge(select,id,window=1)
1
element(setdefaults,edge,nodes=2)
medge(mesh,map,entity="drop")
medge(select,id,window=1)
2
medge(mesh,map,entity="symmetry")
medge(select,id,window=1)
3
medge(mesh,map,entity="piston")
medge(select,id,window=1)
4
medge(mesh,map,entity="right")
END
```

Figure 4.1 - FiGEN portion of a sample FDREAD file

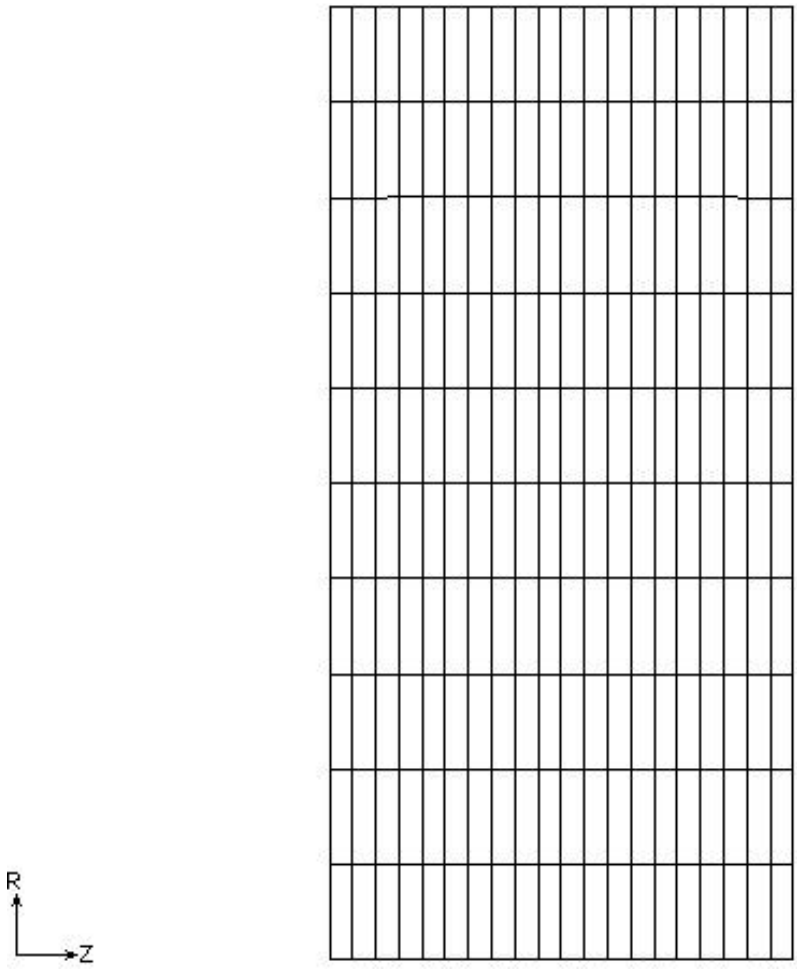

Figure 4.2 - Mesh plot of the simulation at the initial time, zero

rotated to accommodate this. The top plate is on the left side of the diagram, the bottom plate is on the right side, the line of symmetry is at the bottom, and the outer edge is at the top.

Now that the geometry has been defined it is necessary to create the conditions of the problem for the simulation. FiDAP's FiPREP module will be employed for this task. As shown in Figure 4.3 the problem will be defined as axi-symmetric, transient, nonlinear, and free. The axi-symmetric condition is based on the fact that the original geometry of the experiment is a cylinder. A two dimensional model could be used, but both sides of the model would be identical, so it becomes less complicated if the axi-symmetric condition is used. Being that this is a fluids problem with a moving wall it will undoubtedly be both nonlinear and transient. The command 'free' tells the code that we will be implementing the free surface method rather than the filling method used by Hlady.[2] The next lines tell FiDAP how to deal with the pressure as well as telling it not to look for information from a previous simulation; this will be a new job. FiDAP has the added tool of being able to break a long running simulation into several shorter simulations in sequence, a simulation that runs from time 0-10 seconds followed by a simulation running from time 10-20 seconds which would look for information about the previous simulation. Next, the solution command is set to use a segregated solver and the time integration command is set to use the backwards Euler method. This command is also where the starting time is defined, generally zero, the size of our time-step as well as how many time-steps we will require. The time function will be defined to be linear between the two specified points and be as large as or larger than the length of the

```
FIPREP
PROBLEM( ADD, AXI-SYMMETRIC, TRANSIENT, NONLINEAR, FREE )
PRESSURE( ADD, MIXED=1E-17, DISCONTINUOUS )
EXECUTION( ADD, NEWJOB )
SOLUTION( ADD, SEGREGATED=1500, NORMALSTRESS=9, SCHANGE=0, VELCONV=.001,
SURFCONV=.000001 )
RELA
0.3 0.3 0 0.5 0 0.5
TIMEINTEGRATION(ADD, BACKWARD, NSTEPS=$steps, TSTART=$tstart, DT=$DT, FIXED)
TMFUNCTION( ADD, SET=1, NPOINTS=2 )
0 0
1000 1000
DENSITY( ADD, CONSTANT=$rho )
VISCOSITY( ADD, CONSTANT=$MU,CLIP=1E15 )
SURFACETENSION( ADD, SET=1, CONSTANT=$sigma )
ENTITY( ADD, NAME="fluid", FLUID, VOLUME=0 )
ENTITY( ADD, NAME="piston", SURFACE, DEPTH=0, MAPPED, PREFERRED, x=1)
ENTITY( ADD, NAME="drop", SURFACE, DEPTH=0, MAPPED, ANG1=-90, ANG2=90)
ENTITY( ADD, NAME="right", PLOT )
ENTITY( ADD, NAME="symmetry", PLOT )
BCNODE( ADD, SURFACE, ENTITY="piston", CONSTANT=$v, CURVE=1 )
BCNODE( ADD, UZC, ENTITY="piston", CONSTANT=$v )
BCNODE( ADD, VELOCITY, ENTITY="right", ZERO )
BCNODE( ADD, URC, ENTITY="symmetry", ZERO )
BCNODE( ADD, URC, ENTITY="piston", ZERO )
BCNODE( COORDINATE, NODE=1)

END
CREATE( FISOLV )
```

Figure 4.3 - FiPREP portion of a sample FDREAD file

simulation. After that the parameters for the fluid are input, which include the density, *r* in grams/cubic centimeter, while the surface tension will be zero. The viscosity, *n* in poise, will be the value which is varied between each simulation. The last step is to define how the walls will behave and give the boundary conditions. The fluid will be given the property "fluid", the top plate and the outlet, or outflow, will be given the property of "surface" because this will be the free surface. The bottom plate and the line of symmetry will be given the property "plot". The property "plot" does not give the entity any specific properties, rather allows the collection of data on the entity. The boundary conditions will state that the top plate and surface, which must be defined separately, move with a velocity, *v*. The bottom plate and the line of symmetry will have a velocity of zero.

Now the simulation is ready to be run. Executing the FiSOLV module will create the solution so that it is possible to collect and prepare the data. Once the solution has been found the command prompt will be used to tell FiDAP when and where to collect the pressure values that are necessary. It is critical to collect these values at every time-step, with two values being important. It becomes very important to collect data at every time-step because the values will be integrated over time, the more values obtained the more accurate the solution becomes. The important values to collect are the mean pressure and the nodal line pressure. The mean pressure is the average of pressure values that the computer has calculated during it's solution process integrated over the surface area resulting in an average force. The other value, the nodal line pressure, is the pressure at each node on the nodal lines running from the line of symmetry to the outer surface.

Five nodal lines were chosen to be collected, one at the center, two at the plates, and two at the 1/4 distance and 3/4 distance lines. Next, the process by which the most appropriate line is chosen will be discussed, giving results corresponding to the analytical solution.

In order to determine the validity of the simulation it is necessary to test against some experimental data. In order to accomplish this it must first be decided what pressure values are going to be collected from the simulation. One of the options is to use the pressure at one of the walls, preferably the top plate, since this is the one in motion. These values will then be integrated over $2pr$ to give an average pressure over the area in other words the average force. Another option is to use the mean pressure value that the program calculates at every time-step. The third option is to use the pressure values along one of the nodal lines running from the axis of symmetry out to the outlet or droplet. It would again be necessary to integrate these values over $2pr$ to give an average force. Since the mesh is very rough, especially near the walls, where in reality there should be a very fine mesh, it was chosen not to use the pressure values at the wall because there may be fluctuations at the wall that cannot be seen. So the options have now been narrowed to two, the mean pressure and a nodal line integrated over the radius to give and average.

In order to determine which line should be used, which line would be most accurate, a short study was done using five preselected lines. The positions that were chosen to work with were at the top plate, one-fourth, one-half, and three-fourths the distance between top and bottom plates as well as the nodal line along the bottom plate. The study consisted of a liquid bridge with an initial size of approximately 0.4×1.2 cm

with viscosities ranging between 8.0×10^5 and 1.2×10^6. The mesh size would initially be 0.2×0.6 cm and three simulations were made, enough data points to create a line. The viscosities to be used were 8.0×10^5, 1.0×10^6, and 1.2×10^6. The velocity of the wall would be set to 4.0×10^{-4} cm/sec. Once the conditions for the simulation had been set the program was run collecting pressure data at all nodes contained on the five specified lines at every second of the simulation. The next step was to take all of that data and put it into a manageable form. This meant integrating the pressure over $2\pi r$ for every second resulting in an average force along the radius. The next step was to integrate this average force that was just found over time to get an average total force. At each step the numerical results were compared with the analytical results. This step brings attention to any serious deviations that need to be resolved before going farther. Once it was agreed upon that the simulation was, at the least, following closely with the analytical solution and it was acceptable to move further, it was time to decide which of the nodal lines would be best to use. As can be seen in Figure 4.4 the pressures for the nodal lines at the top and bottom plate result in a pressure curve which correlates less with the analytical solution than the other three lines. As described earlier this attributed to the spacing of the mesh, there are possibly fluctuations as well as error accumulation near the surfaces with very few nodes to describe the interactions. Therefore there becomes only three nodal lines to choose from. The choice was made to use the nodal line at the center for two reasons. First, the results were slightly closer to the analytical solution than the other two. Second, if there were to be any fluctuations and/or errors in the computation near the wall, this is the furthest nodal line from the walls and all

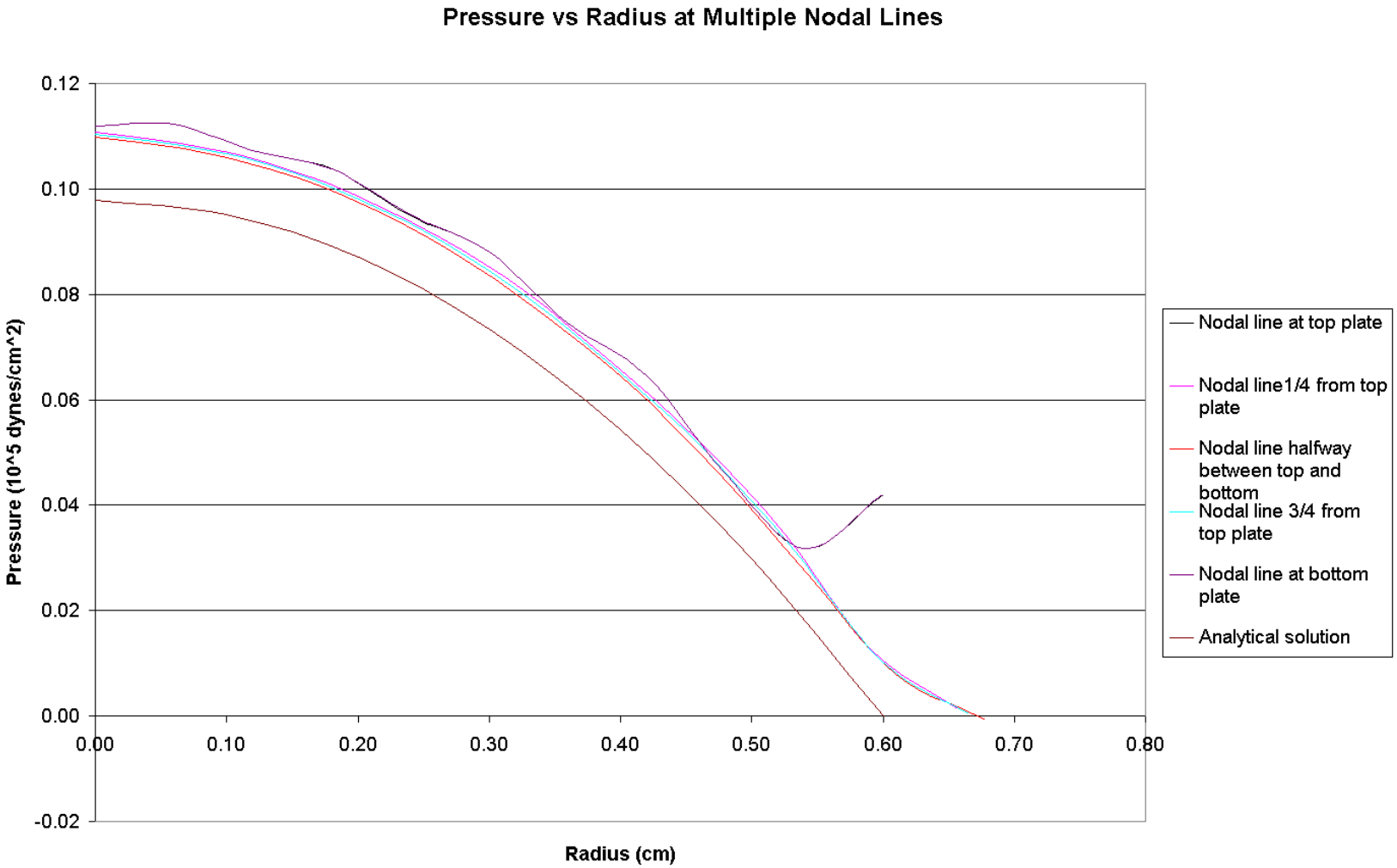

Figure 4.4 - Pressure vs radius diagram for a sample simulation

fluctuations should have dissipated by the time they reach the center. So the assumption that of the five nodal lines, the center line is the best for collecting pressure values for reduction.

The next step will be to determine which pressure values, the mean pressure or the center line pressure, are best to use for comparison to an experimental value. In order to do this, the mean pressure and the center line pressure will be compared to an experimental value. As described previously, the plastometer uses a predefined pressure value and the scientist measures the velocity of the plate. Given this pressure value, there was an experimental baseline with which to compare these two values to. Three slightly different simulations were created with three slightly different viscosities as well while keeping the other parameters constant. Each simulation consisted of a mesh that was 0.2 $\times$ 0.6 cm, density of 4.0 g/cm^3, and surface tension set to zero. The three simulations, named *cenme1*, *cenme2*, and *cenme3*, each had different velocities, 4.06588×10^{-4} cm/sec, 3.66818×10^{-4} cm/sec, and 3.30515×10^{-4} cm/sec respectively. Each of these was subdivided into *a*, *b*, and *c* so that different viscosities could be used with each. These viscosities were 1.0×10^5, 2.5×10^5 and 5.0×10^5 poise respectively. After collecting the mean and center line pressure values at multiple time-steps, the center nodal line pressure values must be integrated over $2pr$. Then both the mean and center nodal average force is integrated over time to get an average total force. Since it was chosen to use three viscosities for each fluid simulated, a graph can be created containing seven lines, the three lines for the center line pressure for each velocity, the three lines for the mean pressure for each velocity, and the experimental line. Looking at Figure 4.5 it is

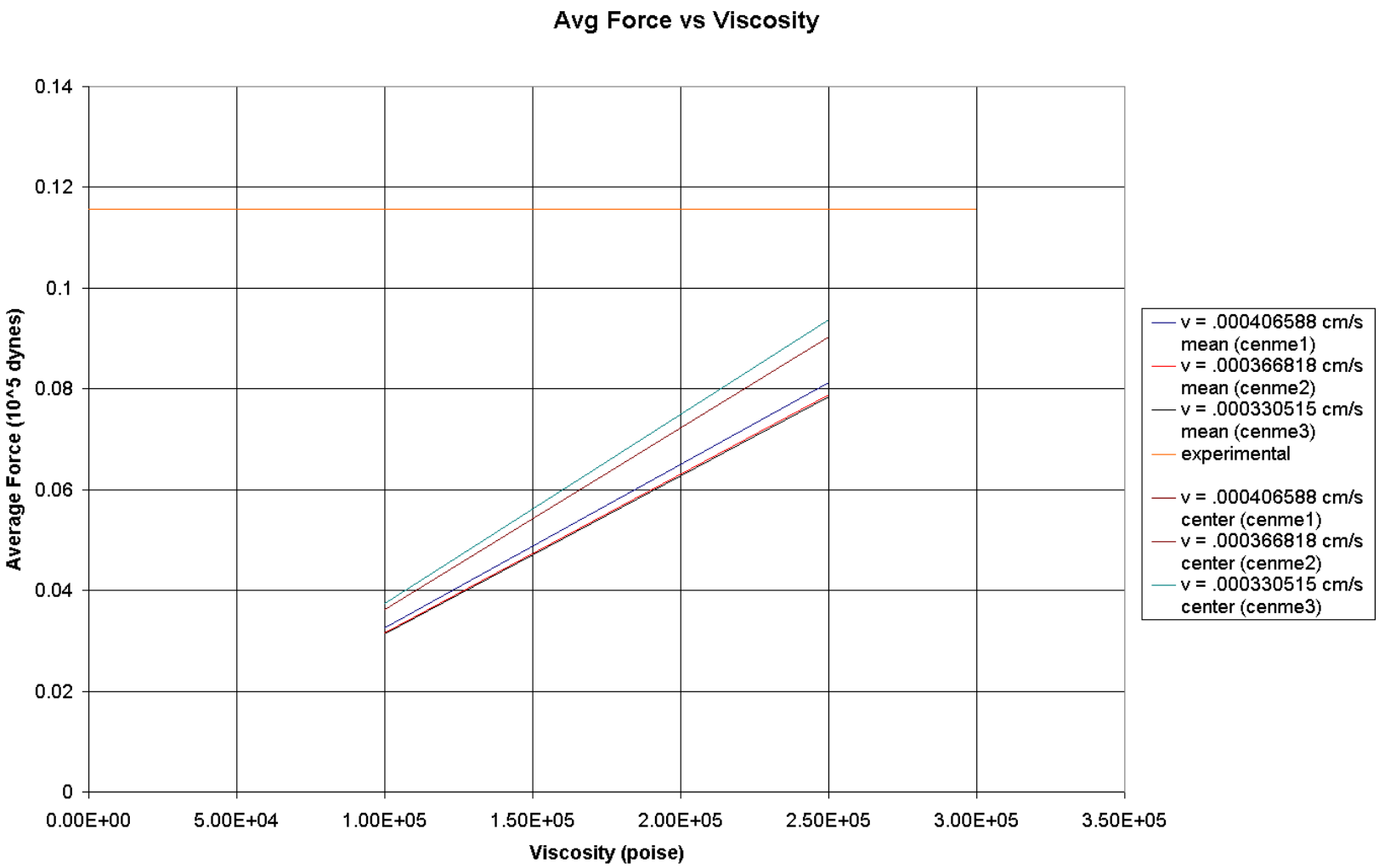

Figure 4.5 - Average force vs. viscosity for a sample simulation

seen that the experimental line does not have the same slope as the numerical lines. The reason for this is that the plastometer uses a constant pressure while the velocity will vary slightly. The numerical solution, as defined here, uses a constant velocity while the pressure may vary. It can also be seen in the figure that the center line pressure values result in average pressures slightly closer to the experimental line than the mean pressure values. This can most probably be attributed to the fact that the computer averages all the pressure values across the entire mesh to get the mean pressure. This would result in the inclusion of some error associated with poor meshing near the walls as well as the inclusion of fluctuations. As a result the assumption that the center line pressure values will be the more appropriate values to use for further work in trying to generate a better model of the viscosity of these fluids, at least until a better mesh is found which can reduce or eliminate the errors near the wall as well as any fluctuations that may propagate inward.

4.3 - Collection and Reduction of the Data

After generating the appropriate simulation, the method that will be used to predict the viscosity of the fluid in question continues as follows. First, use the module FiPREP to collect the data by giving the appropriate commands to collect the pressure values along the centerline and then advance to the next time-step, shown in Figure 4.6. The results are output by FiDAP into the FiOUT file, which is formatted with an enormous amount of extraneous information, mainly text. Two simple C++ programs were created to gather the required information, eliminating all of the text, and formatting

```
FIPOST
LINE(PRESSURE,2NODES)
33,12
MEAN(PRESSURE)
TIMESTEP(STEP=    2     )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
TIMESTEP(STEP=    3     )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
TIMESTEP(STEP=    4     )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
TIMESTEP(STEP=    5     )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
TIMESTEP(STEP=    6     )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
TIMESTEP(STEP=    7     )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
TIMESTEP(STEP=    8     )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
TIMESTEP(STEP=    9     )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
TIMESTEP(STEP=    10    )
LINE(PRESSURE,CURRENT)
MEAN(PRESSURE)
```

Figure 4.6 - FiPOST portion of a sample FDREAD file

the data in a table format in multiple files, one for each time-step, containing the radial position and pressure. These programs are shown in Appendix A-3 and A-4, respectively. The program *fioutmay.exe* takes a single FiOUT file removes all extraneous information and returns a data file with only the vital information, *lisio##.dat*, where ## denotes either letters or numbers indicating which simulation is being run, as well as a file containing only the mean pressure values in a table versus time, *mean.dat*, and a set of files, denoted by *press_###.dat* shown in Figure 4.7, containing the pressure at each time step in a table format versus the radius. The program *press.exe* is a very simple program which creates a single file with a table containing the names and locations of all the *press_###.dat* files for the specific simulation and save the file as *lisio##_press.dat*, shown in Figure 4.8. Mathematica is then employed to integrate the pressure over the radius for each time-step and output the results into a single file. Figure 4.9 shows the Mathematica commands for accomplishing this. Mathematica first imports the *lisio##_press.dat* file and then using a while loop it calls upon each *press_###.dat* file and collects the data, creates an interpolating function, finds the minimum and maximum limits, integrates over those limits and then outputs that value, the average force, to a file called *lisio##_out.dat*. Before Mathematica is used again the *lisio##_out.dat* file must be edited. It contains a list of integrated pressure values so the file must be edited to contain a table with both time and pressure. The file created should be labeled *lisio##_in.dat*. Once the average force at every time-step is in a table format, Mathematica is used again to integrate over time to find an average force applied on the plate, see Figure 4.10 for the Mathematica commands. In this set of commands Mathematica imports the data from the

```
f:/lisio4d/press_001.dat
f:/lisio4d/press_002.dat
f:/lisio4d/press_003.dat
f:/lisio4d/press_004.dat
f:/lisio4d/press_005.dat
f:/lisio4d/press_006.dat
f:/lisio4d/press_007.dat
f:/lisio4d/press_008.dat
f:/lisio4d/press_009.dat
f:/lisio4d/press_010.dat
(etc...)
```

Figure 4.7 - Sample lisio##_press.dat file containing the locations of press_###.dat files

```
0.000000000E+00      0.832281579E+06
0.510119327E-01      0.818222460E+06
0.102023865E+00      0.792638081E+06
0.153035798E+00      0.751477768E+06
0.204047731E+00      0.694562639E+06
0.255059664E+00      0.621973633E+06
0.306071596E+00      0.533801219E+06
0.357083529E+00      0.428549040E+06
0.408095462E+00      0.300060543E+06
0.459107394E+00      0.134966717E+06
```

Figure 4.8 - Sample press_###.dat file containing radius and pressure data

```
n = 0;

temp2 = Import[“F:/lisio2a_press.dat”];

While[(n = n + 1) < 201, file1 = Extract[temp2, {n, 1}]; table1 = Import[file1];

        rec = Interpolation[table1]; limits = First[rec]; maxlimb = Last[First[limits]];

        minlim = First[First[limits]];

        J=NIntegrate[2 * π * x * rec[x], {x, minlim, maxlimb}, MaxRecursion → 1000];
```

Figure 4.9 - Mathematica commands used to integrate the pressure over the radius

```
Import[“F:/lisio2a_in.dat”]

rec = Interpolation[%]

limits = First[rec]

maxlimb = Last[First[limits]]

minlim = First[First[limits]]

J = Nintegrate[rec[x], {x, minim, maxlimb}, MaxRecursion → 1000]

J / (maxlimb - minlim)
```

Figure 4.10 - Mathematica commands used to integrate the average force over time

newly created lisio##_in.dat file and creates an interpolating function. Then it finds the limits of integration and integrates over those limits, outputting the result to the screen. This value is the average force with respect to time. This process creates one of the three points needed to form a line on the plot. After performing this procedure three times, a line can be created and plotted against the value of the force used during the experiment. The point at which these two curves meet determines the predicted viscosity of the fluid.

Chapter 5 - Results and Conclusions

This project was designed to test the ability and accuracy of FiDAP to simulate the compression of a constant velocity liquid bridge for purposes of determining viscosity. From the previous work done by Hlady[2] we resolved that a 'fill' method would not accurately simulate the pressure properties of the experiment. It was then decided that to use the "free surface" method, which would eliminate the necessity of an inflow. This method would allow the movement of walls with a predetermined velocity as well as deform the mesh, which was not accomplished in the 'fill' method.

Due to the fact that the liquid bridge is cylindrical in shape, it was possible to simplify the simulation and use an axi-symmetric model. The initial grid would be rectangular with two plates defined as free surfaces, a line of symmetry, and the outflow, also defined as a free surface. The simulation was set up as described in the previous chapters. The following data was received from NASA Marshall scientists [5], shown in Table 5.1, on four separate plastometer sessions.

Table 5.1 - List of data collected from experiments [5]

i	Diameter (cm)	h_o (cm)	h_f (cm)	rate (cm/sec)	Force (10^5 dynes)	t (sec)	T (°C)
1	1.003	0.0307	0.0304	$8. \times 10^{-7}$	3.950589	3750	500
2	1.03	0.0304	0.0297	1.8×10^{-6}	4.166145	3889	525
3	1.03	0.0297	0.0249	$9. \times 10^{-6}$	4.166145	4556	550
4	1.02	0.0249	0.0228	3.3×10^{-5}	4.085641	636	575

The values the NASA researchers had calculated for the viscosity, using an analytical model, were 2.691×10^{10} poise, 1.099×10^{10} poise, 1.564×10^{9} poise, and 3.406×10^{8} poise, respectively, as shown in Table 5.2. In order to calculate the viscosity it is necessary to create a plot using several viscosities near the supplied experimental viscosity. Using the procedure outlined in the previous chapter to build the input files, run the simulations, collect and reduce the pressure data, the results can be compiled into four plots of pressure vs. viscosity.

As stated in the previous chapter a line was plotted representing the force that was used in the plastometer experiment. Using the intersection point we can predict the viscosity of the fluid being used. In the case of the third numerical solution, shown in Figure 5.1, the predicted viscosity is approximately 2.43×10^{9} poise while the fourth numerical solution, shown in Figure 5.2, shows that the predicted viscosity is 5.071×10^{8} poise. These values are of the same order magnitude as the values predicted by the current analytical model being used in the scientific community. An unknown problem occurred when simulating the plastometer for the first two experiments.

Table 5.2 - Comparison of experimentally found and numerically found viscosities [5]

i	Experimental Viscosity (poise)	Numerical Viscosity
1	2.691×10^{10}	unknown
2	1.099×10^{10}	(2.4×10^{10})*
3	1.546×10^{9}	2.43×10^{9}
4	3.406×10^{8}	5.071×10^{8}

* see discussion

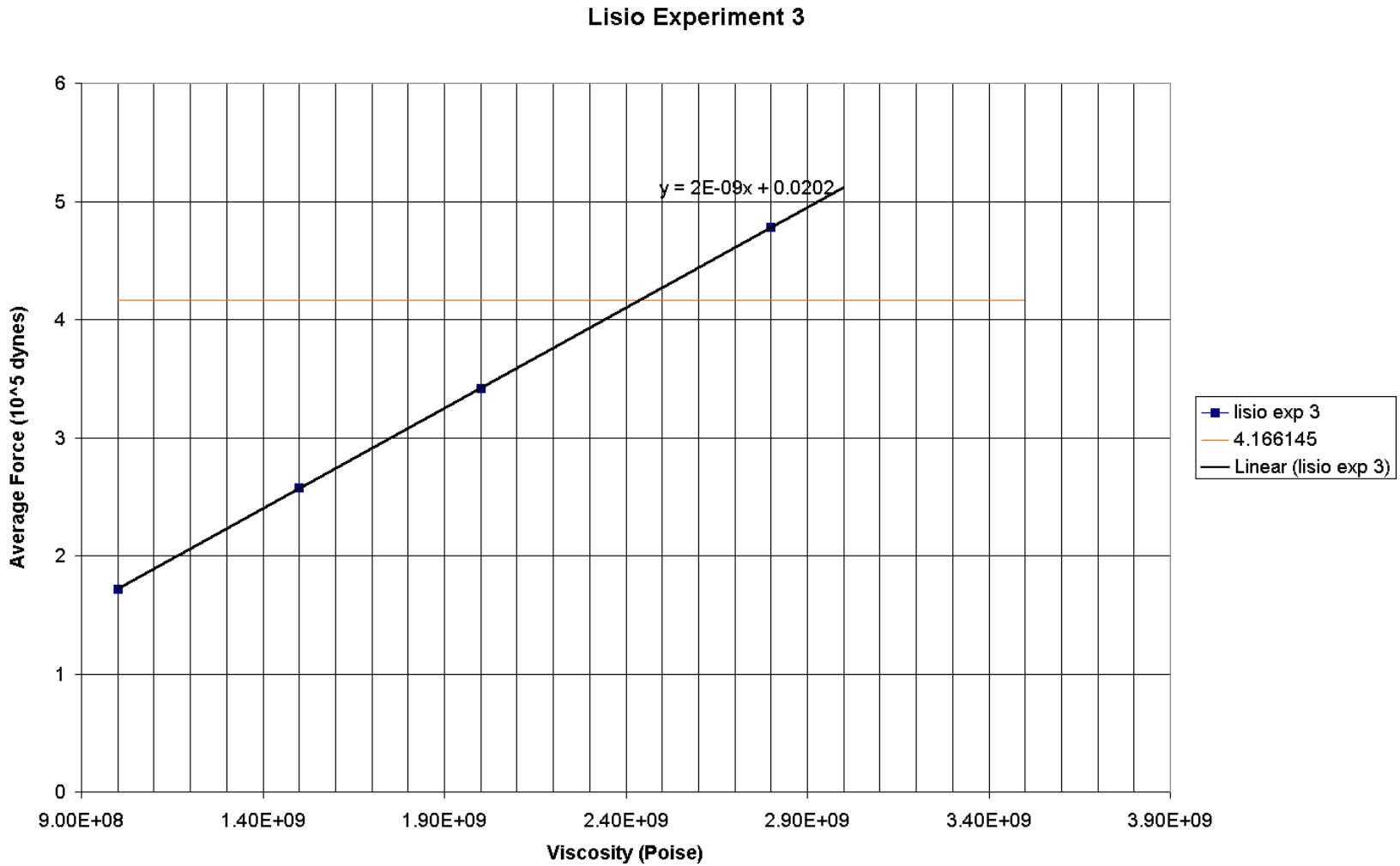

Figure 5.1 - Force vs. viscosity graph for experiment 3

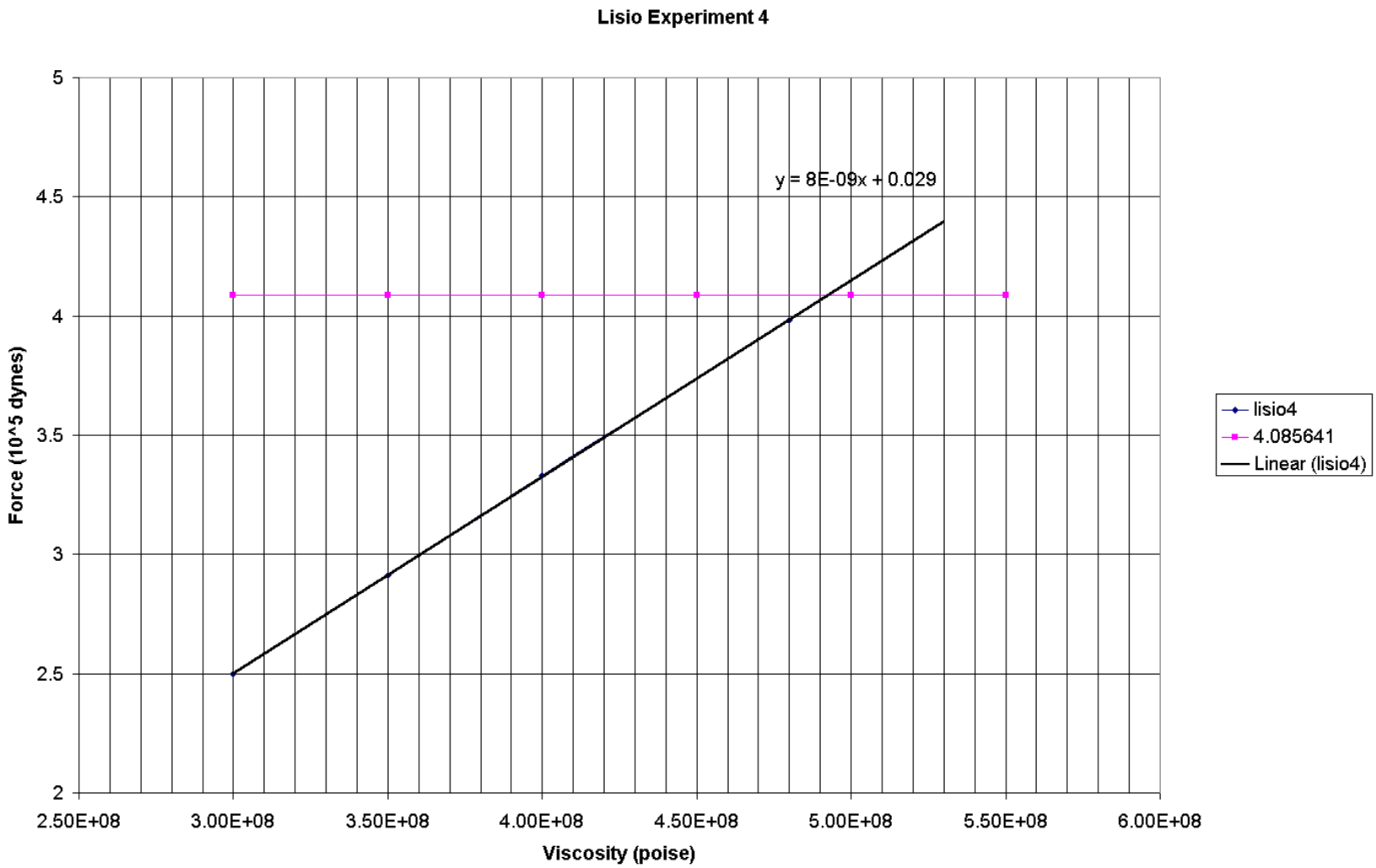

Figure 5.2 - Graph showing force vs. viscosity for experiment 4

The results of the second experiment initially appeared valid after plotting the first three points. A fourth point was needed in order to intersect the value of the force used in the experiment. A deviation from the expected results occurs with this fourth point. Assuming that the point was still possibly valid, a fifth point was generated which had an even larger deviation. As shown in Figure 5.3 the first three points create a straight line with an increasing slope, as expected, but the rest of the points generated break away from the expected and seem to oscillate around 27 Newtons. The results of the first experiment form a curve with a decreasing slope, as shown in Figure 5.4. When this deviation became apparent all input files were checked for errors and the run again, but the results were the same. Several of the clipping values were adjusted in the input files with little or no effect in the results. It is not known as of yet what may be the cause of these errors.

Although the first two experiments do not provide valid data at this juncture, seemingly valid data can be attained from the third and fourth experiments. Shown in Figures 5.5 and 5.6 is the deformation of the free surface of the fluid as well as the motion of the top plate. Figure 5.5 shows a superposition of several different times, while Figure 5.6 shows a mesh plot of both the beginning and ending time used in the simulation, for comparison. Figures 5.7 and 5.8 both show pressure contour plots for the beginning time and ending time respectively.

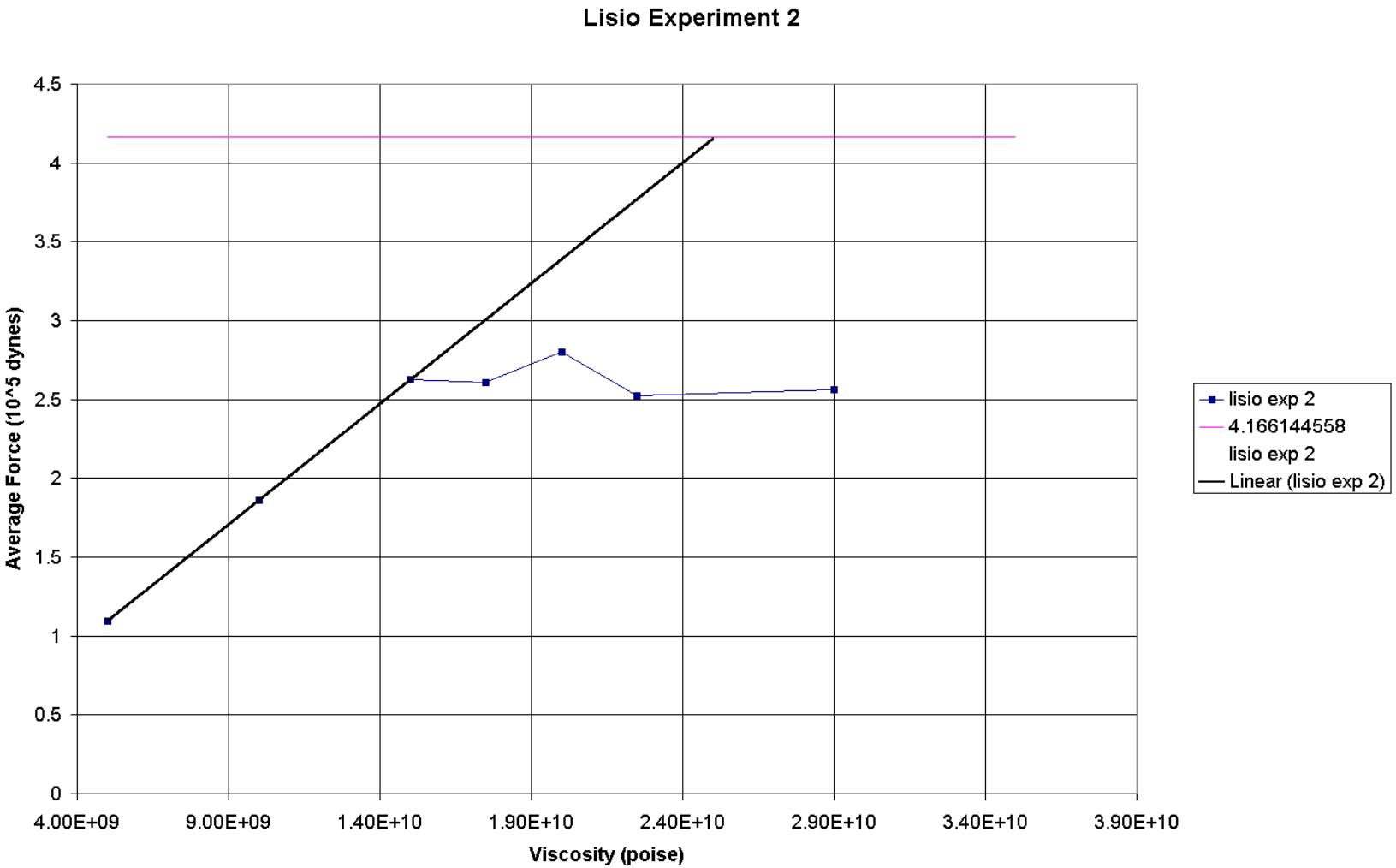

Figure 5.3 - Force vs. viscosity graph for experiment 2

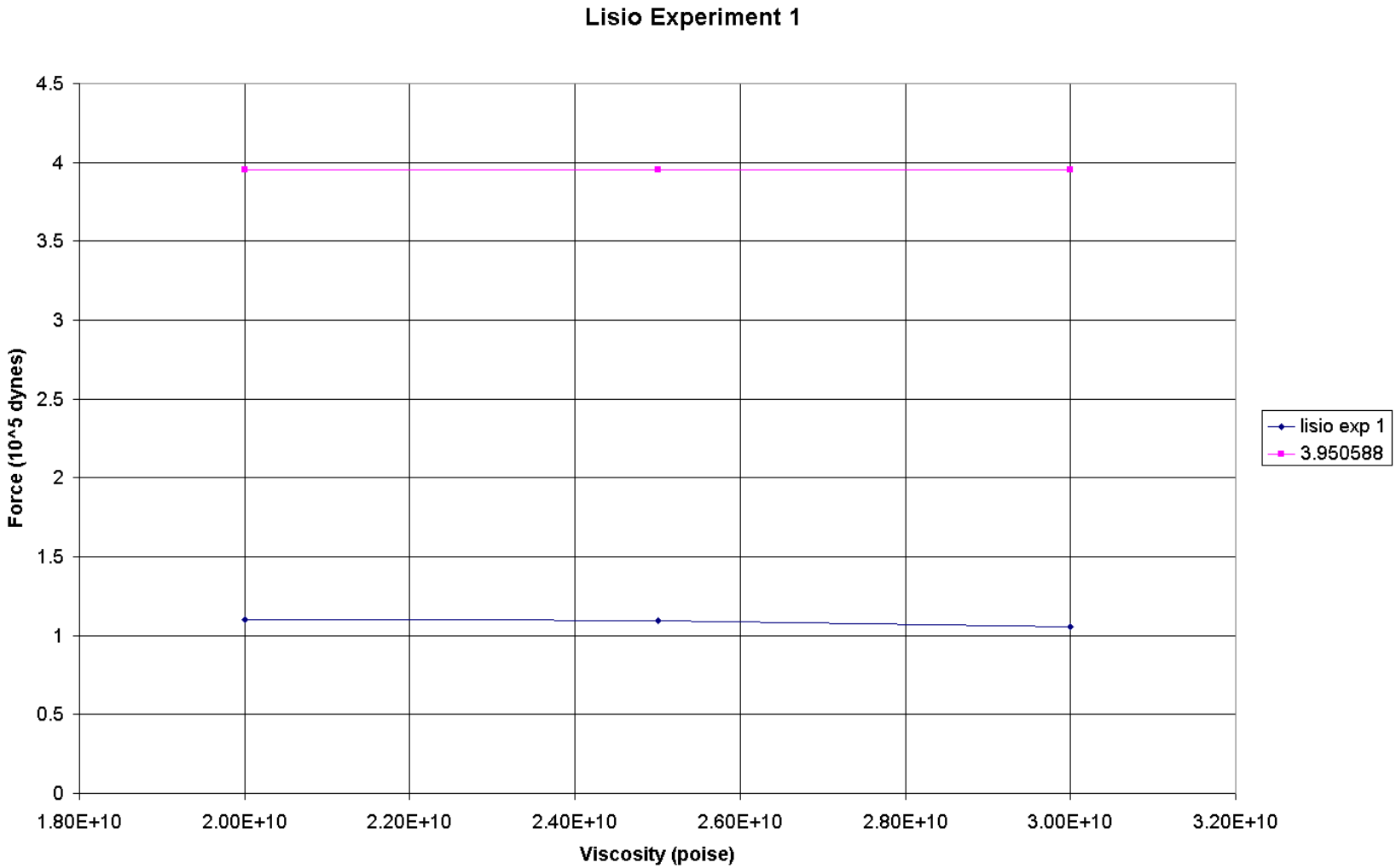

Figure 5.4 - Force vs. viscosity graph for experiment 1

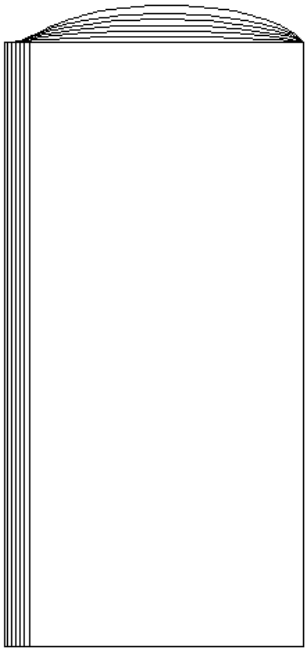

Figure 5.5 - Superposition of multiple edge plots for experiment 4

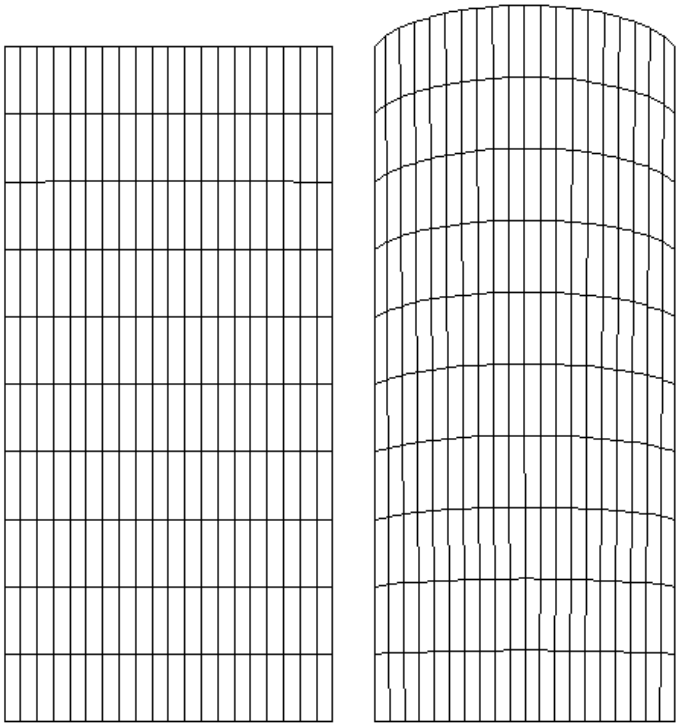

Figure 5.6 - Mesh plots at both first and last time-steps

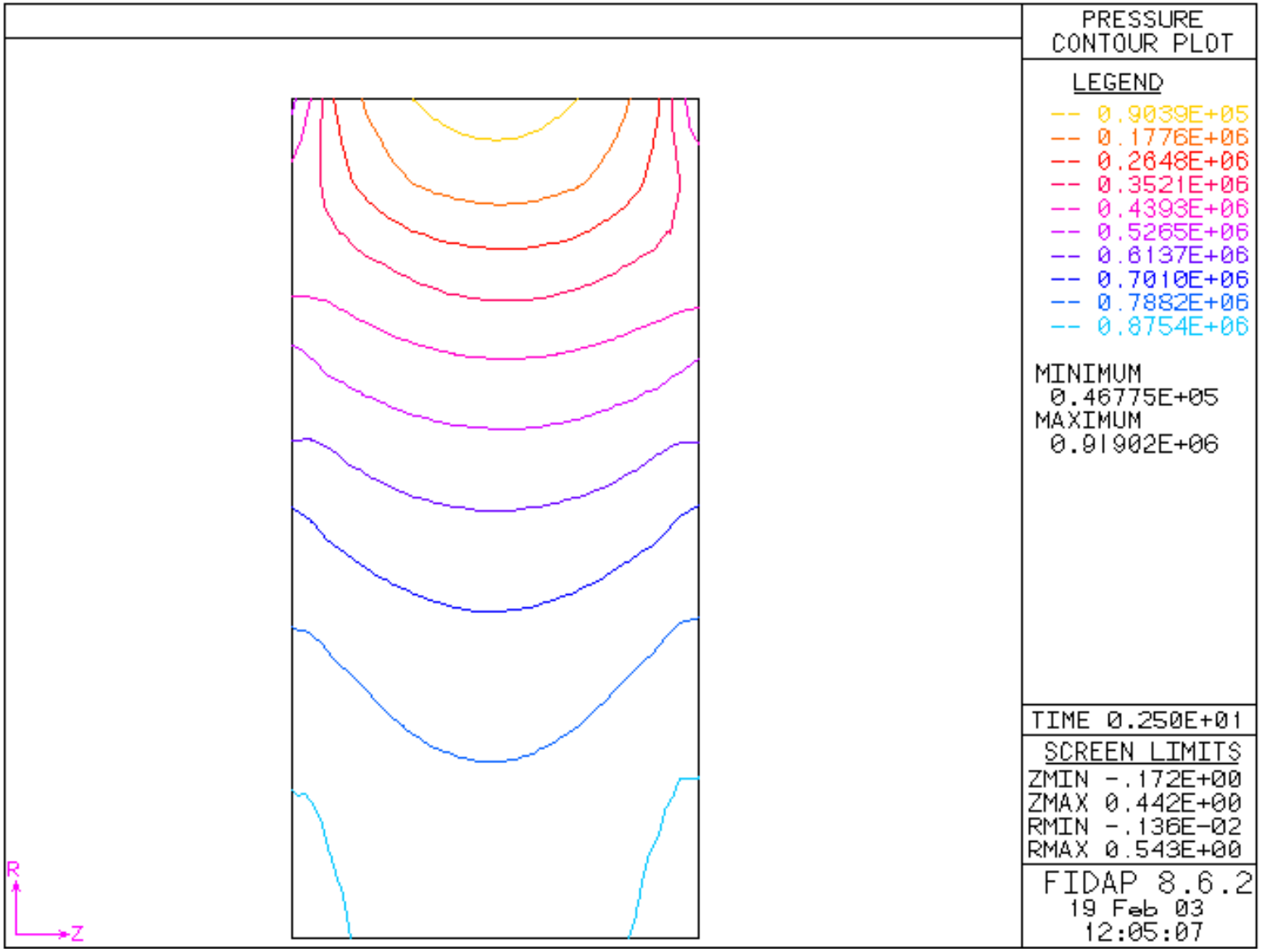

Figure 5.7 - Pressure contour plot for the first time-step

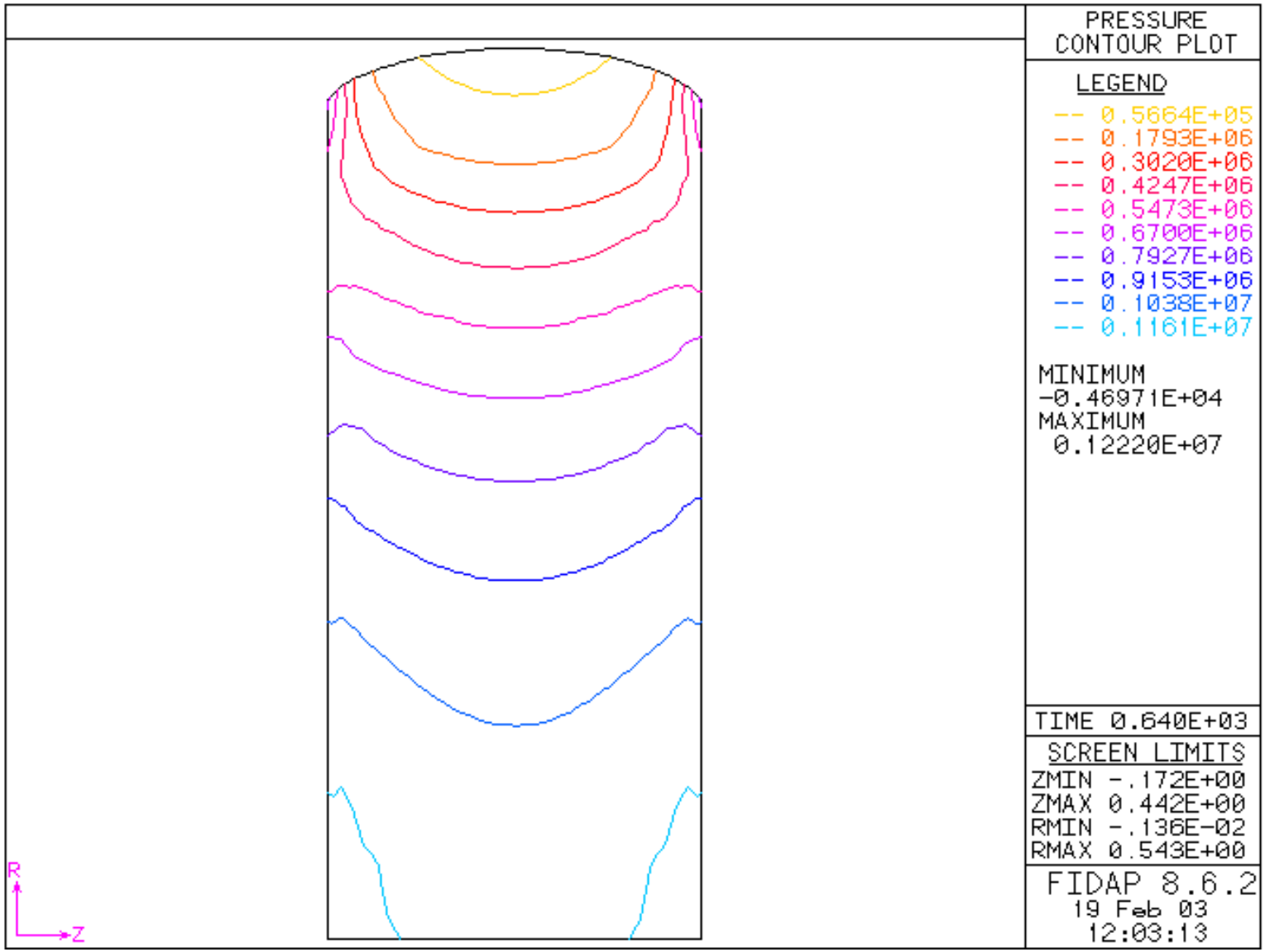

Figure 5.8 - Pressure contour plot for the last time-step, time = 640 seconds

Chapter 6 - Recommendations

From the results obtained for experiments three and four, FiDAP's 'free surface' method appears to be a viable alternative to the current analytical solution being used in the scientific community today. Although more research needs to take place to verify these results. As for the first two experiments, more work needs to be done in order to determine why there is a deviation from the expected results. There are many factors that may be involved in this deviation, such as improper relaxation values or possibly that it may be at the limit of FiDAP's capabilities using such a large viscosity or small velocity. If the cause of the deviation can be worked out, then the next step, before implementation of this system, should be to perfect the mesh increasing the number of nodes near the plates where the pressure is changing both rapidly and dramatically. In the future a better simulation should be developed in which the pressure instead of the velocity is the boundary condition applied to the top plate.

References

List of References

[1] Bird, R.B., Armstrong, R.C., and Hassager, O. *Dynamics of Polymeric Liquids*. John Wiley & Sons, New York, 1987.

[2] Hlady, Sheri, "Numerical Simulation of High Viscous Fluid", Master's Thesis, UTSI, Tullahoma, TN. December 2000.

[3] Dienes, G.J. and Klemm, H.F., "Theory and Application of the Parallel Plate Plastometer". Journal of Applied Physics, Vol. 17, June 1946, pp. 458-471.

[4] Fluid Dynamics International, Inc. FiDAP 8.5 Manuals. Evanston, Il.

[5] Private communications to Dr. Antar from Scientists at NASA Marshall Space Flight Center.

[6] Fontana, E.H. "A Versatile Parallel-Plate Viscometer for Glass Viscosity Measurements to 1000ºC". Ceramic Bulletin. Vol 49, No. 6, 1970, pp. 594-597.

Appendix

A-1
Experiment 4C

```
/.249x.51 2- Dimensional  rho=2.12 sigma=0 mu=4e8, exp04 06/25/02 LISIO4C
$x=.249
$y=.51
$MU=4.0e08
$rho=2.12
$sigma=0
$v=3.3e-05
$tstart=0
$steps=400

FI-GEN (ELEMENT=1,POINT=1,CURVE=1,SURFACE=1,NODE=0,MEDGE=1,
MLOOP=1,MFACE=1,BEDGE=1,SPAVE=1,MSHELL=1,MSOLID=1,
COORDINATE=1 )
point(coordinates,x=0,y=0)
point(coordinates,x=$x,y=0)
point(coordinates,x=0,y=$y)
point(coordinates,x=$x,y=$y)
point(select,id,window=1)
1
2
curve(line,showlabel)
point(select,id,window=1)
3
4
curve(line,showlabel)
point(select,id,window=1)
1
3
curve(line,showlabel)
point(select,id,window=1)
2
4
curve(line,showlabel)
curve(select,id,window=1)
1
3
2
4
mface(wireframe,edg1cnt=1,edg2cnt=1,edg3cnt=1,edg4cnt=1)
curve(select,id,window=1)
2
```

```
1
medge(frstlast,intervals=20,ratio=0.000000,2ratio=0.000000,pcentr=0.000000)
curve(select,id,window=1)
3
4
medge(frstlast,intervals=10,ratio=0.000000,2ratio=0.000000,pcentr=0.000000)
mface(select,id,window=1)
1
mface(mesh,map,entity="fluid")
medge(select,id,window=1)
1
element(setdefaults,edge,nodes=2)
medge(mesh,map,entity="drop")
medge(select,id,window=1)
2
medge(mesh,map,entity="symmetry")
medge(select,id,window=1)
3
medge(mesh,map,entity="piston")
medge(select,id,window=1)
4
medge(mesh,map,entity="right")
END

FIPREP
PROBLEM( ADD, AXI-SYMMETRIC, TRANSIENT, NONLINEAR, FREE )
PRESSURE( ADD, MIXED=1E-17, DISCONTINUOUS )
EXECUTION( ADD, NEWJOB )
SOLUTION( ADD, SEGREGATED=1500, NORMALSTRESS=9, SCHANGE=0,
VELCONV=.001, SURFCONV=.000001 )
RELA
0.3 0.3 0 0.5 0 0.5
TIMEINTEGRATION(ADD, BACKWARD, NSTEPS=$steps, TSTART=$tstart,
DT=10, FIXED)
TMFUNCTION( ADD, SET=1, NPOINTS=2 )
0 0
1000 1000
DENSITY( ADD, CONSTANT=$rho )
VISCOSITY( ADD, CONSTANT=$MU,CLIP=1E15 )
SURFACETENSION( ADD, SET=1, CONSTANT=$sigma )
ENTITY( ADD, NAME="fluid", FLUID, VOLUME=0 )
ENTITY( ADD, NAME="piston", SURFACE, DEPTH=0, MAPPED, PREFERRED,
x=1)
ENTITY( ADD, NAME="drop", SURFACE, DEPTH=0, MAPPED, ANG1=-90,
ANG2=90)
```

```
ENTITY( ADD, NAME="right", PLOT )
ENTITY( ADD, NAME="symmetry", PLOT )
BCNODE( ADD, SURFACE, ENTITY="piston", CONSTANT=$v, CURVE=1 )
BCNODE( ADD, UZC, ENTITY="piston", CONSTANT=$v )
BCNODE( ADD, VELOCITY, ENTITY="right", ZERO )
BCNODE( ADD, URC, ENTITY="symmetry", ZERO )
BCNODE( ADD, URC, ENTITY="piston", ZERO )
BCNODE( COORDINATE, NODE=1)

END
CREATE( FISOLV )
```

A-2
Experiment 4D

```
/.249x.51 2- Dimensional  rho=2.12 sigma=0 mu=4.8e8, exp04 07/11/02
LISIO4D
$x=.249
$y=.51
$MU=4.8e08
$rho=2.12
$sigma=0
$v=3.3e-05
$tstart=0
$DT=10
$steps=256

FI-GEN ( ELEMENT=1,POINT=1,CURVE=1,SURFACE=1,NODE=0,MEDGE=1,
MLOOP=1,MFACE=1,BEDGE=1,SPAVE=1,MSHELL=1,MSOLID=1,
COORDINATE=1 )
point(coordinates,x=0,y=0)
point(coordinates,x=$x,y=0)
point(coordinates,x=0,y=$y)
point(coordinates,x=$x,y=$y)
point(select,id,window=1)
1
2
curve(line,showlabel)
point(select,id,window=1)
3
4
curve(line,showlabel)
point(select,id,window=1)
1
3
curve(line,showlabel)
point(select,id,window=1)
2
4
curve(line,showlabel)
curve(select,id,window=1)
1
3
2
4
mface(wireframe,edg1cnt=1,edg2cnt=1,edg3cnt=1,edg4cnt=1)
```

```
curve(select,id,window=1)
2
1
medge(frstlast,intervals=20,ratio=0.000000,2ratio=0.000000,pcentr=0.000000)
curve(select,id,window=1)
3
4
medge(frstlast,intervals=10,ratio=0.000000,2ratio=0.000000,pcentr=0.000000)
mface(select,id,window=1)
1
mface(mesh,map,entity="fluid")
medge(select,id,window=1)
1
element(setdefaults,edge,nodes=2)
medge(mesh,map,entity="drop")
medge(select,id,window=1)
2
medge(mesh,map,entity="symmetry")
medge(select,id,window=1)
3
medge(mesh,map,entity="piston")
medge(select,id,window=1)
4
medge(mesh,map,entity="right")
END

FIPREP
PROBLEM( ADD, AXI-SYMMETRIC, TRANSIENT, NONLINEAR, FREE )
PRESSURE( ADD, MIXED=1E-17, DISCONTINUOUS )
EXECUTION( ADD, NEWJOB )
SOLUTION( ADD, SEGREGATED=1500, NORMALSTRESS=9, SCHANGE=0,
VELCONV=.001, SURFCONV=.000001 )
RELA
0.3 0.3 0 0.5 0 0.5
TIMEINTEGRATION(ADD, BACKWARD, NSTEPS=$steps, TSTART=$tstart,
DT=$DT, FIXED)
TMFUNCTION( ADD, SET=1, NPOINTS=2 )
0 0
1000 1000
DENSITY( ADD, CONSTANT=$rho )
VISCOSITY( ADD, CONSTANT=$MU,CLIP=1E15 )
SURFACETENSION( ADD, SET=1, CONSTANT=$sigma )
ENTITY( ADD, NAME="fluid", FLUID, VOLUME=0 )
ENTITY( ADD, NAME="piston", SURFACE, DEPTH=0, MAPPED, PREFERRED,
x=1)
```

```
ENTITY( ADD, NAME="drop", SURFACE, DEPTH=0, MAPPED, ANG1=-90,
ANG2=90)
ENTITY( ADD, NAME="right", PLOT )
ENTITY( ADD, NAME="symmetry", PLOT )
BCNODE( ADD, SURFACE, ENTITY="piston", CONSTANT=$v, CURVE=1 )
BCNODE( ADD, UZC, ENTITY="piston", CONSTANT=$v )
BCNODE( ADD, VELOCITY, ENTITY="right", ZERO )
BCNODE( ADD, URC, ENTITY="symmetry", ZERO )
BCNODE( ADD, URC, ENTITY="piston", ZERO )
BCNODE( COORDINATE, NODE=1)

END
CREATE( FISOLV )
```

A-3
Fioutmay.cpp

Function: Removes all unnecessary text from the FiOUT output file and creates multiple press_###.dat files containing pressure and radius information

```
#include <stdio.h>
#include <math.h>
#include <fstream.h>
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
#include <string.h>
typedef char EndLine[601];
typedef char part1[18];
typedef char part2[17];
typedef char String14[15];
typedef char cmpto[40];
typedef char Blank[2];

struct inputdata
{       double time;
        int step;
        double H1,H2,volume, intpress, meanpress;     };
typedef inputdata data_array[1801];
void main()
{       data_array data;
        String14 filename;
        String14 make;
        String14 make2;
        String14 output;
        cout << "What FIOUT file do you want edited?" << endl;
        cin >> filename;
        cout  << "What name do you want for the dat file (name.dat)?" <<endl;
        cin >> make;
        ifstream din;
        ofstream dout;
        din.open(filename);
        dout.open(make);
        EndLine line;
        EndLine line2;
        cmpto Integrated = " I N T E G R A T E D  V A L U E";
        cmpto Volume = " T O T A L  V O L U M E/A R E A";
        cmpto Mean = " M E A N  V A L U E   ";
        cmpto Time = " TIMESTEP ";
```

```
        cmpto point = "     POINT         ";
        cmpto extime = " ***    0.000 COMMAND EXECUTION TIME ";
        cmpto hr = "h/r";
        part1 first;
        part2 value;
        part2 h1,h2;
        int i=0;
        dout << " TIMESTEP =    0    TIME =   0.0000000" << endl;
        while (din)
        {       din.getline(line, 601);
                if (strncmp(line,Integrated,31)==0)
                {       dout << line << endl;                   }
                if (strncmp(line,Volume,31)==0)
                {       dout << line << endl;                   }
                if (strncmp(line,Mean,19)==0)
                {       dout << line << endl;                   }
                if (strncmp(line,Time,9)==0)
                {       dout << line << endl;                   }
                if (strncmp(line,point,18)==0)
                {       din.getline(line, 601);
                        din.getline(first, 16);
                        i = 1;
                        while (strncmp(first,point,4)==0 || i==1)
                        {       din.getline(value,17);
                                dout << value << "  ";
                                if (i==1)
                                        strcpy(h1,value);
                                else
                                        strcpy(h2,value);
                                din.getline(line, 601);
                                dout << line << endl;
                                din.getline(first,16);
                                i++;    }
                        dout << endl << "h/r1 = " << h1 << endl << "h/r2 = " << h2 <<
endl;   }}
        dout.close();
        din.close();
        din.open(make);
        int l=0;
        int k;
        while (din)
        {       k=0;
                din.getline(line,601);
                if ( strncmp(line,Integrated,5) == 0)
                {       din.getline(line,601);
```

```
                    if(strncmp(line, Volume, 10) == 0)
                    {       din.getline(line,37);
                            din >> data[l].meanpress;
                            din.getline(line,601);          }
                    l++;    }
            sprintf(output, "press_%03d.dat",l);
            dout.open(output);
            while( strncmp(line,Time,5) !=0 && strncmp(line,"h/r",3) !=0 &&
strncmp(line,Integrated,5)!=0 && strncmp(line,Volume,5)!=0 &&
strncmp(line,Mean,5)!=0 && strncmp(line2,line,15) !=0)
            {       strcpy(line2,line);
                    dout << line << endl;
                    din.getline(line,601);
                    dout << line << endl;          }
            dout.close();  }
        dout.open("mean.dat");
        int m=0;
        double del;
        cout << "What is delta t?" << endl;
        cin >> del;
        double t=del;
        while(m<l)
        {       dout << t << "                  " << data[m].meanpress << endl;
                m++;
                t=t+del;        }
        dout.close();
        din.close();
        int num=1;
        int count=0;
        din.open(make);
        while(din)
        {       sprintf(output, "press_%03d.dat",num);
                dout.open(output);
                din.getline(line,601);
                if(strncmp(line,Time,5) == 0)
                {       din.getline(line,601);
                        while (strncmp(hr,line,3) != 0)
                        {       strcpy(line2,line);
                                dout << line << endl;
                                din.getline(line,601);
                                count++;       }}
                if (count >=11)
                {       num++;
                        count = 0;     }
                dout.close();  }}
```

A-4
Press.cpp

Function : Creates a file which lists the locations of the press_###.dat files output by the program in appendix A-3.

```
#include <stdio.h>
#include <math.h>
#include <fstream.h>
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
#include <string.h>
typedef char String14[15];
typedef char String60[61];
void main()
{       String14 filename, file, directory, number;
        String60 append;
        int n=1, end_number;
        cout << "What filename do you want to create? file.dat" << endl;
        cin >> filename;
        cout << "What directory are the files in?" << endl;
        cin >> directory;
        cout << "What is the name of the file ???_###.dat?  Leave off the ###.dat
        only give ???_" << endl;
        cin >> file;
        cout << "What is the ending file number?" << endl;
        cin >> end_number;
        ofstream dout;
        dout.open(filename);
        while (n<=end_number)
        {       strcpy(append,directory);
                strcat(append,file);
                sprintf(number,"%03d",n);
                strcat(append, number);
                strcat(append, ".dat");
                dout << append << endl;
                n=n+1;                  }
        dout.close(); }
```

Vita

Brian C. Robinson was born in Freeland, MD on November 5th, 1978. After finishing high school in 1996 he went on to receive a Bachelor of Science degree in Physics at Salisbury State University, now Salisbury University, in the spring of 2000. After enrolling at UTSI he complete this thesis in order to receive a Master of Science degree in Engineering Science in the Spring of 2003. He hopes to also receive a Master of Science degree in Aerospace engineering very shortly and then find a nice job to settle into.